

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **05346922 A**

(43) Date of publication of application: **27 . 12 . 93**

(51) Int. Cl

G06F 15/20
G06F 9/06

(21) Application number: **04153946**

(22) Date of filing: **15 . 06 . 92**

(71) Applicant: **PFU LTD**

(72) Inventor: **OKUDAIRA YUMIKO**
SHINDO TOKUMI
WATARI AKIKO
YAMADA TOSHIAKI
KOIKE TADASHI

(54) **DYNAMIC REFLECTING METHOD FOR
OPERATION ENVIRONMENT**

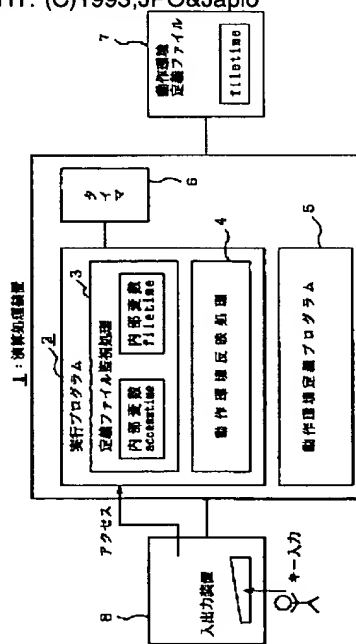
(57) Abstract:

PURPOSE: To reduce the load and to speedily and dynamically reflect operation environment on an execution program by monitoring access and reflecting the operation environment on the execution program only when the access interval is longer than a specific time and an operation environment definition file is updated as to the dynamic reflecting method which dynamically reflects the operation environment on the execution program.

CONSTITUTION: This method is equipped with a definition file monitoring process 3 for monitoring whether or not the interval of access to the execution program exceeds the specific time and an operation environment definition file 7 wherein the operation environment is set; and the access is monitored through the definition file monitoring process 3 and if it is made evident that the operation environment definition file 7 is updated by referring to the operation environment definition file 7 when a state wherein access is attained longer than the specific time after a last access time is detected, the updated operation environment is taken out of the operation environment

definition file 7 to update the operation environment of the execution program, thereby performing processes under the new operation environment.

COPYRIGHT: (C)1993,JPO&Japio



Japanese Unexamined Patent Application Publication

No. 5-346922

[Title of the Invention] DYNAMIC REFLECTING METHOD OF
OPERATING ENVIRONMENT

[0019] Fig. 1 is a configuration diagram of an embodiment of the present invention. In Fig. 1 a processing unit 1 carries out various processes. The processing unit 1 comprises an execution program 2 performing Kana-Kanji conversion and the like, an operating environment defining program 5 setting an operating environment in an operating environment definition file 7, a timer 6 and the like.

[0020] The execution program 2 is a program conducting various kinds of processing such as Kana-Kanji conversion, and comprises, in this embodiment, a definition file monitoring and an operating environment reflecting 4.

[0021] The definition file monitoring 3 monitors access from the input/output unit 8 to the execution program 2 to see whether or not the interval between accesses has exceeded a prescribed period of time. This access monitoring is to determine the necessity of reflecting the operating environment onto the execution program 2, when the execution program 2 is accessed, on condition that the difference between the current time fetched with reference to the

timer 6 and the time fetched from the internal variable "accesstime" storing the last access time is larger than a certain value, and the time fetched from the internal variable "filetime" storing the last update time of the operating environment definition file 7 and the time fetched from the variable "filetime" of the operating environment definition file 7 are not equal to each other, and the operating environment has been updated.

[0022] The internal variable "accesstime" is a variable storing the last access time. The internal variable "filetime" is a variable storing the time upon the last update of the operating environment definition file 7.

[0023] The operating environment reflecting 4 is to set the operating environment fetched from the operating environment definition file 7 in the execution program 2, and causes execution of processing in the thus set new operating environment.

[0024] The operating environment definition program 5 defines (sets, updates) an operating environment specified by the operator from the input-output unit 8 or the like in the operating environment definition file 7. In this definition, the variable "filetime" is previously updated time (see Fig. 3).

[0025] The timer 6 is to count the current time. The operating environment definition file 7 is a file storing definitions of the

current environments, and has a variable "filetime" storing the time when making a definition (setting, updating). The execution file monitoring 3 compares the time of the internal variable "filetime" and the time of this variable "filetime", and they differ from each other. If different, the operating environment is determined to have been updated.

[0026] The input/output 8 conducts various operations of input and output, comprising, for example, a keyboard and a display. The operations with the configuration shown in Fig. 1 will now be described in detail in accordance with the sequence shown in the flowchart of Fig. 2.

[0027] In Fig. 2, initial values of the internal variable "filetime" and the internal variable "accesstime" are set in S1. Initial values are set with reference to the operating environment definition file 7 and from the timer to the current time. For example, initialization is conducted as follows:

Initial variable filetime ← f0 (time f0 set for the variable filetime of the operating environment definition file 7), and

Internal variable accesstime ← a0 (current time of timer 6)

[0028] S2 is a step of waiting for an access from the operator. S3 is a step of determining whether or not an operator access has occurred. This is accomplished by the operator from the keyboard

which is the input/output unit 8 shown in Fig. 1 by, for example, entering a character string to be subjected to Kana-Kanji conversion to determine whether or not an access to the execution program 2 has occurred. In the case of YES, the process proceeds to S4 and the subsequent steps. In the case of NO, suggesting occurrence of no access, the process returns to S2.

[0029] S4 is a step of acquiring the current time a1. This step comprises acquiring the current time a1 from the timer 6. In S5, it is determined whether or not a certain period of time has elapsed from the time a0 stored in the internal variable accesstime to the current time a1 acquired in S4.

This is determined by:

$a1 - a0 > \text{certain period of time}$

In the case of YES, suggesting that a certain period of time has elapsed from the last access to the current time, the process goes to S6. In the case of NO, on the other hand, revealing that no certain period of time has elapsed from the last access to the current time, reflecting the operating environment in the execution program is discontinued. The internal accesstime is updated to the current time (internal variable accesstime ← a1) in S10, and processing is carried out in S12, to return to S2.

[0030] S6 is a step of acquiring the definition file time f1 of the

operating environment definition file 7. This is accomplished by fetching the time f1 stored in the variable filetype of the operating environment definition file 7 shown in Fig. 1.

[0031] In S7, it is determined whether or not the definition file time f1 of the operating environment definition file 7 has been updated. This is the determination of $f1 \neq f0$. More specifically, this is to compare the time f0 of the variable filetype of the operating environment definition file 7 when fetching an operating environment from the operating environment definition file 7 last time and reflecting the same in the execution program 2, and the time f1 of the variable filetype of the current operating environment definition file 7, and to check if these values are not equal (updated). If YES, which suggests update of the operating environment definition file 7, the process transfers to S8. If NO, on the other hand, suggesting that the operating environment definition file 7 is not updated, the interval accesstime is updated into a current time (internal variable accesstime \leftarrow a1) in S11, and processing is executed in S12, returning to S2.

[0032] In S8, the internal variable accesstime is updated into the current time, and the internal variable filetype is updated into the current time. In other words, updating is accomplished as follows:

Internal variable accesstime \leftarrow a1 (current time)

Internal variable filetime ← f1 (updated time f1 of the operating environment definition file 7).

These steps S4 to S8, S10 and S11 are carried out by the definition file monitoring 3 shown in Fig. 1.

[0033] In S9, the execution program 2 reflects the operating environment definition file 7, i.e., fetches an operating environment after updating environment from the operating environment definition file 7, sets the same into an executable state, and executes processing under this operating environment in S12, for example, carries out Kana-Kanji conversion. S9 is accomplished by the operating reflection4.

[0034] As a result of the above, only when the interval between accesses from the input/output unit 8 to the execution program 2 is over a certain period of time (YES in S5), and the operating environment definition file 7 has been updated (YES in S7), the updated operating environment is fetched from the operating environment definition file 7; the fetched operating environment is reflected onto the execution program; and processing is executed on the basis of this updated operating environment. Accordingly, when the execution program 2 is frequently accessed by the operator via the input/output unit 8 such as the keyboard, the load is alleviated by not referring to the operating environment file 7. Since a

certain period of time is necessary for updating the operating environment definition file 7, only when an access is made after the lapse of the certain period of time expected to be necessary for updating, it is checked up whether or not the operating environment has been updated with reference to the operating environment definition file 7. When updated, the operating environment is quickly reflected in the execution program.

FIG. 1

CONFIGURATION DIAGRAM OF FIRST EMBODIMENT OF THE
INVENTION

- 1: PROCESSING UNIT
- 2: EXECUTION PROGRAM
- 3: DEFINITION FILE MONITORING
- INTERNAL VARIABLE accesstime
- 4: OPERATING ENVIRONMENT REFLECTION
- 5: OPERATING ENVIRONMENT DEFINITION PROGRAM
- 6: TIMER
- 7: OPERATING ENVIRONMENT DEFINITION FILE
- 8: INPUT/OUTPUT UNIT
- (1) ACCESS

(2) KEY-INPUT

FIG. 2

REFLECTING FLOWCHART OF OPERATING ENVIRONMENT
DEFINITION FILE OF THE INVENTION

S1: INITIALIZATION OF INTERNAL VARIABLES filetime AND
 accesstime

 SET WITH REFERENCE TO filetime IN DEFINITION FILE;

 SET WITH REFERENCE TO CURRENT TIME FROM TIMER;

 filetime ← f0

 accesstime ← a0

S2: WAITING FOR ACCESS FROM OPERATOR

S3: HAS OPERATOR ACCESS OCCURRED?

S4: ACQUIRE CURRENT TIME (a1)

S5: HAS CERTAIN TIME ELAPSED FROM accesstime?

 (a1 - a0 . certain time)

S6: ACQUIRE DEFINITION FILE TIME f1

S7: HAS DEFINITION FILE TIME BEEN UPDATED? (f1 ≠ f0)

S8: UPDATE accesstime (accesstime ← a1)

 update filetime (filetime ← f1)

S9: EXECUTION PROGRAM REFLECTS DEFINITION FILE

S10: UPDATE accesstime (accesstime ← a1)

S11: UPDATE accesstime (accesstime \leftarrow a1)

S12: EXECUTE PROCESSING

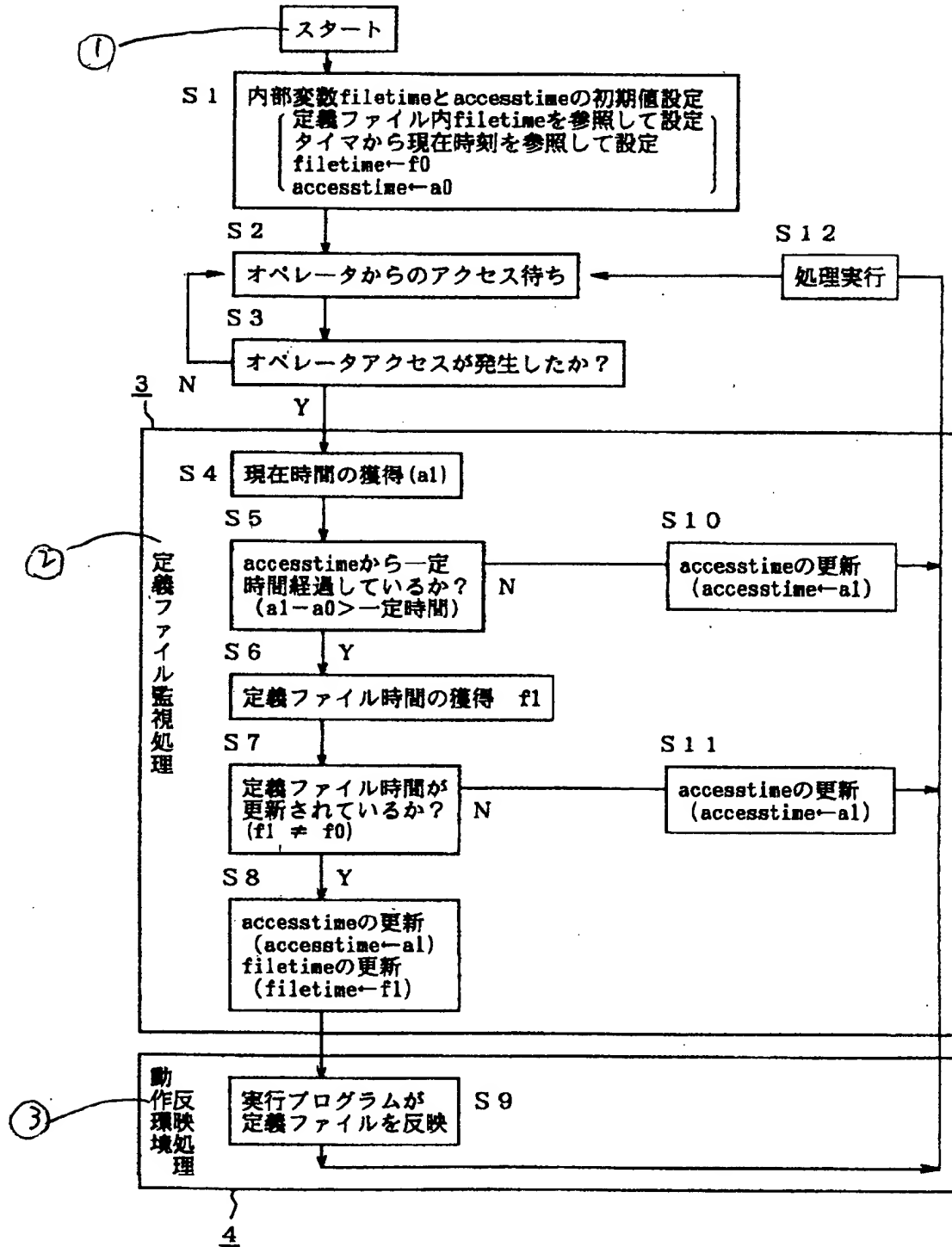
(1) START

(2) DEFINITION FILE MONITORING

(3) OPERATING ENVIRONMENT REFLECTION

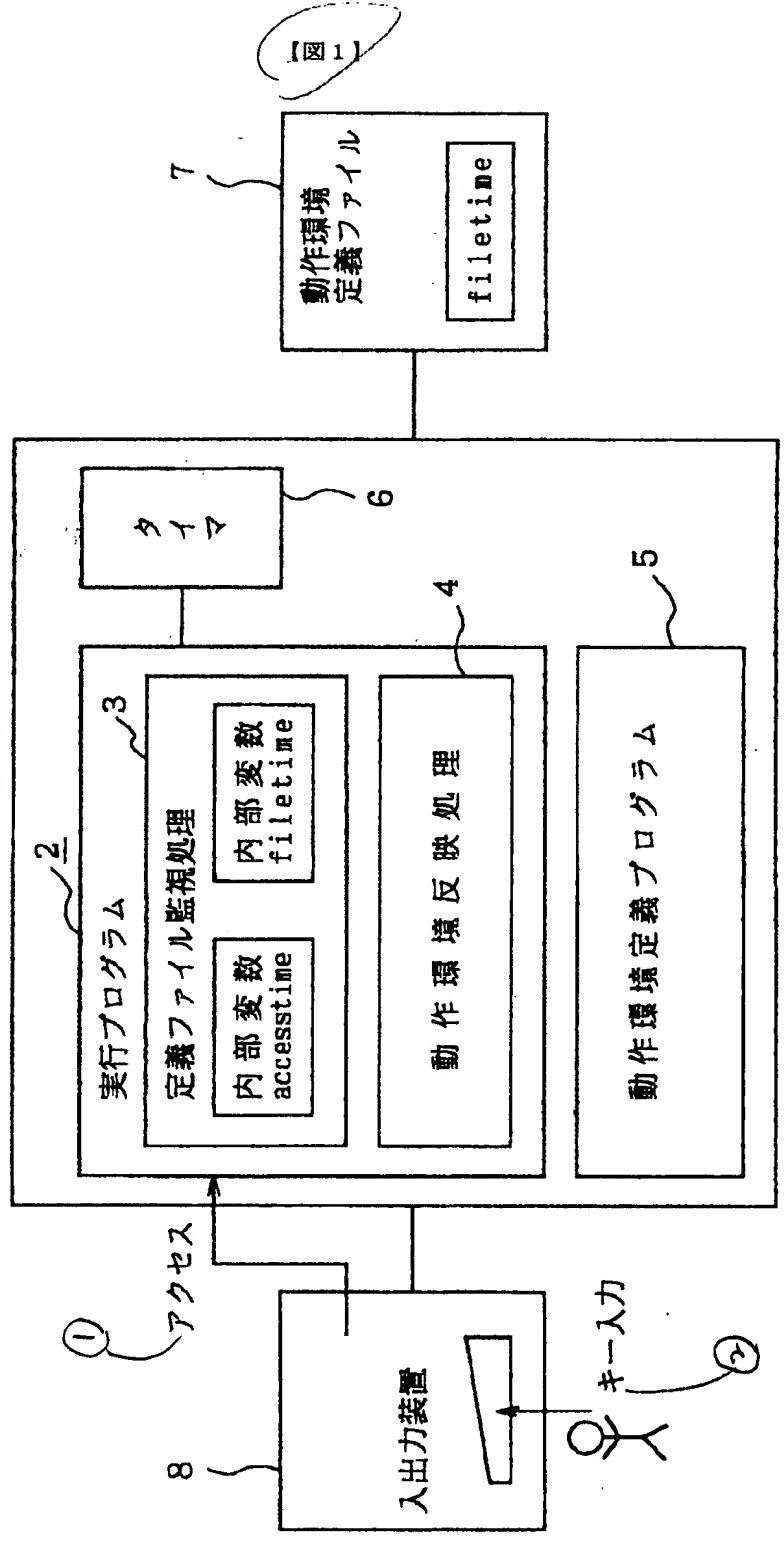
【図2】

本発明の動作環境定義ファイルの反映フローチャート



本発明の1実施例構成図

1: 演算処理装置



【特許請求の範囲】

【請求項1】動作環境を動的に実行プログラムに反映する動的反映方法において、

実行プログラムへのアクセスの間隔が所定時間以上経過したかを監視する定義ファイル監視処理(3)と、動作環境を設定する動作環境定義ファイル(7)とを備え、

上記定義ファイル監視処理(3)がアクセスを監視し、アクセスが前回のアクセス時間から所定時間以上経過していることを検出したときに、上記動作環境定義ファイル(7)を参照して更新されていると判明したときに当該動作環境定義ファイル(7)から更新後の動作環境を取り出し、実行プログラムの動作環境を更新し、新たな動作環境のもとで処理を実行するように構成したことを特徴とする動作環境の動的判定方法。

【請求項2】上記定義ファイル監視処理(3)が前回のアクセス時間を記憶する内部変数accesstimeおよび上記動作環境定義ファイル(7)の前回の更新時間を記憶する内部変数filetimeを持つと共に、当該動作環境定義ファイル(7)に更新時間を記憶する変数filetimeを持ち、

この動作環境定義ファイル(7)を更新したときに変数filetimeを更新しておき、

定義ファイル監視処理(3)がアクセスを監視し、現在のアクセス時間と上記内部変数accesstimeの時間とを比較および更新し、所定時間以上経過していると判明したときに、上記動作環境定義ファイル(7)の変数filetimeを参照して内部変数filetimeと異なると判明したときに当該動作環境定義ファイル(7)から更新後の動作環境を取り出して実行プログラムの動作環境に反映および内部変数filetimeを更新するように構成したことを特徴とする請求項1記載の動作環境の動的判定方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、動作環境を動的に実行プログラムに反映する動的反映方法に関するものである。

【0002】

【従来の技術】従来、かな漢字変換システムなどは、動作環境であるキーバインドや変換方法などをオペレータが任意に設定し、操作し易いようにキーボード中の任意のキーに文字、記号を割り当てたり、変換した結果、得られる漢字の範囲(JIS第1水準、JIS第2水準など)を変更したり、更に専門分野に応じて辞書を切り替えたりしている。

【0003】また、動作環境の設定は、プログラムの動作中も行える。この場合には、更新した動作環境をかな漢字変換システムに反映させるためには、プログラムを一旦停止させ、再起動して動作環境を取り込む。

【0004】このように動作環境を更新したときにプロ

グラムを一旦停止し、再起動させて反映することは不便であるので、プログラムの動作中に動作環境定義ファイルが更新されたときに、プログラムを停止することなく反映することが望まれ、これを達成するために以下のような方法が考えられる。

【0005】(1) 第1の方法は、オペレータからのアクセス毎に動作環境定義ファイルが更新されているか否かを常にチェックし、更新されていた場合にプログラムにこの更新された動作環境の反映を行う。

10 【0006】(2) 第2の方法は、一定周期に割込で動作環境定義ファイルが更新されているか否かをチェックし、更新されていた場合にプログラムにこの更新された動作環境の反映を行う。

【0007】(3) 第3の方法は、動作環境定義ファイルをチェックした時間を記憶しておき、オペレータからのアクセス毎にこの記憶した時間を参照して前回から一定時間以上経過していたら動作環境定義ファイルが更新されているか否かをチェックし、更新されていた場合にプログラムにこの更新された動作環境の反映を行う共に、

20 【0008】

【発明が解決しようとする課題】従来は、プログラムの動作中に動作環境定義ファイルが更新されたときに、プログラムにその更新後の動作環境を反映するために、上記(1)、(2)、(3)の方法が考えられるが、これらの方法について、図5、図6、図7に示すように、時間31で動作環境定義ファイルの更新が発生した場合、それぞれ以下の問題がある。

30 【0009】上記(1)の第1の方法は、図5に示すように、動作環境定義ファイルの更新をすぐに補足し、時間32で動作環境をプログラムに反映できるが、13回の動作環境定義ファイルへのアクセスが必要となり、負荷が大きいという問題がある。

【0010】上記(2)の第2の方法は、図6に示すように、一定時間5で割り込みを発生させた場合、時間35で動作環境をプログラムに反映できるが、8回の動作環境定義ファイルへのアクセスが必要となり、アクセス回数が減って負荷は少し減少するが、プログラムへの反映が時間35となって遅れてしまうという問題がある。

40 【0011】上記(3)の第3の方法は、図7に示すように、動作環境定義ファイルをアクセスしてチェックしてから次のチェックまでの時間差分5とした場合、時間32で動作環境をプログラムに反映できるが、7回の動作環境定義ファイルへのアクセスが必要となり、アクセス回数が減って負荷は少し減少するが、まだ7回もの動作環境定義ファイルへのアクセスが行われてしまう問題がある。

【0012】本発明は、これらの問題を解決するため、アクセスを監視してアクセス間隔が所定時間以上であつて、かつ動作環境定義ファイルが更新されているときに

のみ動作環境を実行プログラムに反映し、負荷を少なくしてかつ迅速に動作環境を動的に実行プログラムへの反映を可能にすることを目的としている。

【0013】

【課題を解決するための手段】図1を参照して課題を解決するための手段を説明する。図1において、定義ファイル監視処理3は、アクセスの間隔が所定時間以上経過したか監視するものであって、前回のアクセス時間を記憶する内部変数accesstimeおよび動作環境定義ファイル7の前の更新時間を記憶する内部変数filetimeを持つものである。

【0014】動作環境定義ファイル7は、動作環境を保持するものであって、更新時間を記憶する変数filetimeを内部に持つものである。

【0015】

【作用】本発明は、図1に示すように、定義ファイル監視処理3がアクセスを監視し、アクセスが前回のアクセス時間から所定時間以上経過していることを検出したときに、動作環境定義ファイル7を参照して更新されていると判明したときに当該動作環境定義ファイル7から更新後の動作環境を取り出し、実行プログラムの動作環境を更新し、新たな動作環境のもとで処理を実行するようにしている。

【0016】また、定義ファイル監視処理3がアクセスを監視し、現在のアクセス時間と内部変数accesstimeの時間とを比較および更新し、所定時間以上経過していると判明したときに、動作環境定義ファイル7の変数filetimeを参照して内部変数filetimeと異なると判明したときに当該動作環境定義ファイル7から更新後の動作環境を取り出して実行プログラムに反映すると共に内部変数filetimeを更新し、新たな動作環境のもとで処理を実行するようにしている。

【0017】従って、アクセスを監視してアクセス間隔が所定時間以上であって、かつ動作環境定義ファイル7が更新されているときにのみ動作環境を実行プログラムに反映することにより、負荷を少なくして動作環境を動的に実行プログラムに迅速に反映することが可能となる。

【0018】

【実施例】次に、図1から図4を用いて本発明の実施例の構成および動作を順次詳細に説明する。

【0019】図1は、本発明の1実施例構成図を示す。図1において、演算処理装置1は、各種演算処理を行うものである。この演算処理装置1は、かな漢字変換などを行う実行プログラム2、動作環境定義ファイル7に動作環境を設定する動作環境定義プログラム5、およびタイマ6などから構成されるものである。

【0020】実行プログラム2は、各種処理を行うプログラムであって、例えばかな漢字変換を行うプログラムであり、ここでは、定義ファイル監視処理3および動作

環境反映処理4などから構成されるものである。

【0021】定義ファイル監視処理3は、入出力装置8などからの実行プログラム2に対するアクセスを監視し、アクセスの間隔が所定時間以上経過したか監視するものである。このアクセスの監視は、アクセスがあったときに、タイマ6を参照して取り出した現時間と、前回のアクセス時間を記憶する内部変数accesstimeから取り出した時間との差が一定以上であって、かつ動作環境定義ファイル7の前の更新時間を記憶する内部変数filetimeから取り出した時間と、動作環境定義ファイル7の変数filetimeから取り出した時間とを比較し、等しくなくて動作環境が更新されており、動作環境を実行プログラム2に反映する必要があるか判定するものである。

【0022】内部変数accesstimeは、前回のアクセス時間を記憶する変数である。内部変数filetimeは、前回の動作環境定義ファイル7を更新したときの時間を記憶する変数である。

【0023】動作環境反映処理4は、動作環境定義ファイル7から取り出した動作環境を実行プログラム2に設定し、この設定後の新たな動作環境で処理を実行させるものである。

【0024】動作環境定義プログラム5は、入出力装置8などからオペレータによって指示された動作環境を、動作環境定義ファイル7に定義（設定、更新）するものである。この際、変数filetimeを更新した時間に設定しておく（図3参照）。

【0025】タイマ6は、現時間を計数するものである。動作環境定義ファイル7は、動作環境の定義を格納するファイルであって、定義（設定、更新）したときの時間を記憶する変数filetimeを持つものである。実行ファイル監視処理3が内部変数filetimeの時間とこの変数filetimeの時間とを比較し、異なるときに動作環境が更新されていると判定する。

【0026】入出力装置8は、各種入出力を行うものであって、例えばキーボードやディスプレイである。次に、図2のフローチャートに示す順序に従い、図1の構成の動作を詳細に説明する。

【0027】図2において、S1は、内部変数filetimeと、内部変数accesstimeの初期値設定する。初期値は、動作環境定義ファイル7を参照して設定、およびタイマから現在時間を参照して設定する。例えば
内部変数filetime←f0（動作環境定義ファイル7の変数filetimeに設定されている時間f0）
内部変数accesstime←a0（タイマ6の現時間a0）と初期設定する。

【0028】S2は、オペレータからのアクセス待ちする。S3は、オペレータアクセスが発生したか判別する。これは、図1の入出力装置8であるキーボードからオペレータが例えばかな漢字変換する文字列を入力して実行プログラム2に対するアクセスが発生したか判別す

10

20

30

40

50

5

る。YESの場合には、S4以降に進む。一方、NOの場合には、アクセスが発生しないのでS2に戻る。

【0029】S4は、現在時間a1の獲得を行う。これは、現在の時間a1をタイマ6から獲得する。S5は、内部変数accesstimeに記憶している時間a0から、S4で獲得した現在の時間a1までの間に一定時間が経過しているか判別する。これは、

$a1 - a0 > \text{一定時間}$

か判別する。YESの場合には、前回にアクセスした時間から現在の時間までの間に一定時間が経過していたので、S6に進む。一方、NOの場合には、前回にアクセスした時間から現在の時間までの間に一定時間が経過していないので、動作環境を実行プログラムに反映することを止め、S10で内部accesstimeを現時間に更新(内部変数accesstime←a1)し、S12で処理を実行し、S2に戻る。

【0030】S6は、動作環境定義ファイル7の定義ファイル時間f1の獲得を行う。これは、図1の動作環境定義ファイル7の変数filetimeに記憶されている時間f1を取り出す。

【0031】S7は、動作環境定義ファイル7の定義ファイル時間f1が更新されているか判別する。これは、 $f1 \neq f0$ か判別する。即ち前回に動作環境定義ファイル7から動作環境を取り出して実行プログラム2に反映したときの当該動作環境定義ファイル7の変数filetimeの時間f0と、現在の動作環境定義ファイル7の変数filetimeの時間f1とを比較し、等しくない(更新されている)か判別する。YESの場合には、動作環境定義ファイル7が更新されていると判明したので、S8に進む。一方、NOの場合には、動作環境定義ファイル7が更新されていないと判明したので、S11で内部accesstimeを現時間に更新(内部変数accesstime←a1)し、S12で処理を実行し、S2に戻る。

【0032】S8は、内部変数accesstimeを現時間に更新、および内部変数filetimeを現時間に更新する。即ち、

内部変数accesstime←a1 (現時間)

内部変数filetime←f1 (動作環境定義ファイル7の更新時間f1)

とする。これらS4からS8、S10、S11は図1の定義ファイル監視処理3が行う。

【0033】S9は、実行プログラム2が動作環境定義ファイル7を反映、即ち動作環境定義ファイル7から更新後の動作環境を取り出し、実行し得る状態に設定し、S12でこの動作環境のもとで処理を実行、例えばかな漢字変換を行う。ここで、S9は動作反映処理4が行う。

【0034】以上によって、入出力装置8から実行プログラム2に対するアクセス間隔が一定時間以上であって(S5のYES)、かつ動作環境定義ファイル7の動作

6

環境が更新されていた場合(S7のYES)にのみ、動作環境定義ファイル7から更新された動作環境を取り出し、実行プログラム2に反映し、この更新された動作環境をもとに処理を実行する。これらにより、入出力装置8である例えばキーボードなどからオペレータにより実行プログラム2が頻繁にアクセスされているときは動作環境ファイル7を参照しないようにして負荷を軽減し、一方、動作環境定義ファイル7を更新するには一定時間必要であるので、この更新に要すると予想される一定時間を経過した後にアクセスがあった場合にのみ動作環境定義ファイル7を参照して動作環境が更新されているか否かをチェックし、更新されている場合に動作環境を実行プログラムに迅速に反映する。

【0035】図3は、本発明の動作環境定義ファイルの変更フローチャートを示す。これは、オペレータからの指示に従って、図1の動作環境定義ファイル7を更新するときの手順である。

【0036】図3において、S21は、オペレータが定義変更コマンドを押下する。これは、図1の入出力装置8であるキーボード上でオペレータが特定の定義変更コマンドのキーを押下し、図1の動作環境定義プログラム5を起動する。

【0037】S22は、選択可のパターンを表示する。これは、S21の定義変更コマンドの押下に対応して、例えばかな漢字変換の動作環境として、現在、あるキーを押下したとき、“.”と表示されるが他の選択可のパターン“.” “、”などを表示する。

【0038】S23は、選択したか判別する。YESの場合には、S24に進む。NOの場合には、S22で選択可のパターンを表示し、選択するまで待機する。S24は、動作環境定義ファイル7を新たな動作環境に更新する。

【0039】S25は、動作環境定義ファイル7の変数filetimeを現在の時間に更新する。以上によって、オペレータが定義変更コマンドを押下して実行プログラム2の動作環境を保存する動作環境定義ファイル7の動作環境を更新する。この動作環境定義ファイル7の動作環境を更新した状態では、実行プログラム2の動作環境は更新されず、次の起動を待たなければ更新されない。実行プログラム2の動作中に動的に動作環境を更新するには、既述した図2の手順によって動的に更新する。

【0040】図4は、本発明の具体例説明図を示す。これは、図3のフローチャートに従って動作環境定義ファイル7が更新されたときに、図2のフローチャートに従い当該動作環境定義ファイル7から動作環境を動的に実行プログラム2に反映するときの具体例である。ここで・オペレータのアクセスタイミングは、図1でオペレータが入出力装置8であるキーボードからキー入力して“かな”を入力して実行プログラム2をアクセスするタイミングである。

【0041】・時間軸は、時間0から時間35をここでは刻む。

・accesstimeの更新点と、時間差分は、定義ファイル監視処理3が持つ内部変数accesstimeの更新点と、時間差分を示す。

【0042】・実行プログラムの動作は、図1の実行プログラム2の動作を示す。

次に、動作を説明する。

(1) 時間0：オペレータのアクセスタイミングであるので、実行プログラム2の定義ファイル監視処理3がファイルチェックする。この初期状態では、動作環境が更新されていないので、動作環境を実行プログラム2に反映しない(図2のS1)。

【0043】(2) 時間2：オペレータのアクセスタイミングであるので、実行プログラム2の定義ファイル監視処理3がアクセスタイムの経過時間について一定時間5を越えていないので(図2のS5のNO)、内部変数accesstimeを現時間に更新し、動作環境定義ファイル7を参照しない。

【0044】(3) 同様に、時間5、時間7、時間9、時間11についても一定時間4を越えていないので(図2のS5のNO)、内部変数accesstimeを現時間にそれぞれ更新し、動作環境定義ファイル7を参照しない。

【0045】(4) 時間16：実行プログラム2の定義ファイル監視処理3がアクセスタイムの経過時間について一定時間4を越えているので(図2のS5のYES)、動作環境定義ファイル7をチェックするが動作環境が更新されていないので(図2のS7のNO)、動作環境の反映を行わない。

【0046】(5) 時間18、時間20、時間22、時間25、時間27についても一定時間4を越えていないので(図2のS5のNO)、内部変数accesstimeを現時間にそれぞれ更新し、動作環境定義ファイル7を参照しない。

【0047】(6) 時間32：実行プログラム2の定義ファイル監視処理3がアクセスタイムの経過時間について一定時間4を越えているので(図2のS5のYES)、動作環境定義ファイル7をチェックすると動作環境が、動作環境定義プログラム5によって時間31のときに更新されていたので(図2のS7のYES)、動作環境を取り出して実行プログラム2に反映する。

【0048】以上の手順により、ファイルチェック(動作環境定義ファイル7を参照して動作環境が更新されているかのチェック)を行う回数がここでは、3回で済み、しかも動作環境定義ファイル7をオペレータが更新操作を行うと、オペレータアクセスタイミングがここでは一定時間4を越えてしまう特質を利用し、迅速に動作環境定義ファイル7の更新を検出し、更新後の動作環境を動的に迅速に実行プログラム2に反映し、反映した後

の動作環境で処理を実行することが可能となる。

【0049】尚、図1から図4を用いて説明した構成をかな漢字変換システムに使用し、動作環境を更新する例としては以下の場合がある。

(1) 変換辞書の動作環境を更新する場合：個別辞書である、人名辞書、地名辞書、専門用語辞書などを必要に応じて追加したり、削除したり、切り替えたりを行う。

【0050】(2) かな漢字変換する文字列の動作環境を更新する場合：

・複文節解析

複文節変換時に意味優先か学習優先かを選択する。

【0051】・複合語

変換時に複合語優先か否かを選択する。

・数詞変換

数詞変換を行うか否かを選択する。

【0052】・JIS漢字

変換対象文字を常用漢字のみにするか、JIS第2水準以外の文字にするか、全ての文字にするかを選択する。

【0053】(3) 読み文字の入力の動作環境を更新する場合：

・カンマの処理

カンマのキーを打鍵した時、カンマを表示するか、読点を表示するかを選択する。

【0054】・ピリオドの処理

ピリオドのキーを打鍵した時、ピリオドを表示するか、句点を表示するかを選択する。

【0055】(4) 変換する時のスタイルの動作環境を更新する場合：

・次変換時カーソル位置

変換する時のカーソルの位置を選択する。

【0056】・変換文字列の属性の動作環境を更新する場合：変換する時の文字列の属性(下線を引くか、反転させるかなど)を選択する。

【0057】・候補群選択画面の動作環境を更新する場合：候補群を出す時の画面スタイルを選択する。

(5) キーバインドの動作環境を更新する場合：

・変換のためのキーをキーボード上のどのキーに割り当てるかなどを変更する。

【0058】

【発明の効果】以上説明したように、本発明によれば、アクセスを監視してアクセス間隔が所定時間以上であつて、かつ動作環境定義ファイル7が更新されているときにのみ動作環境を実行プログラムに反映する構成を採用しているため、負荷を少なくして動作環境を動的に実行プログラムに迅速に反映することができる。

【図面の簡単な説明】

【図1】本発明の1実施例構成図である。

【図2】本発明の動作環境定義ファイルの反映フローチャートである。

【図3】本発明の動作環境定義ファイルの変更フローチャートである。

【図4】本発明の具体例説明図である。

【図5】従来技術の説明図（その1）である。

【図6】従来技術の説明図（その2）である。

【図7】従来技術の説明図（その3）である。

【符号の説明】

1：演算処理装置

2：実行プログラム

3：定義ファイル監視処理

4：動作環境反映処理

5：動作環境定義プログラム

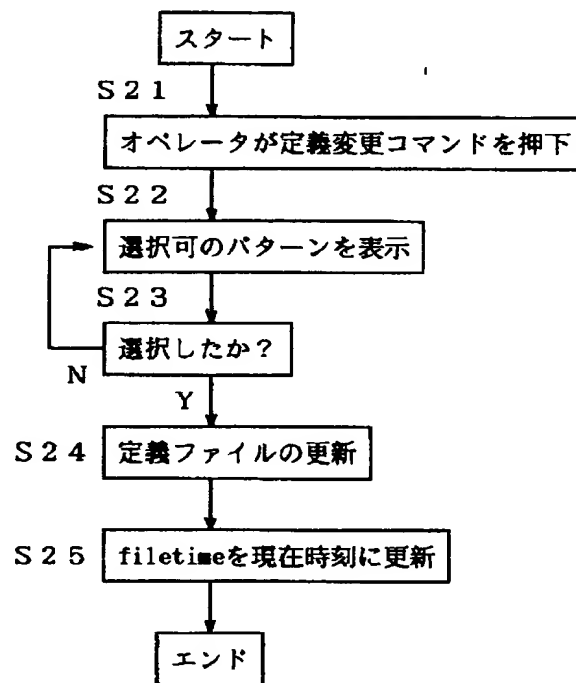
6：タイマ

7：動作環境定義ファイル

8：入出力装置

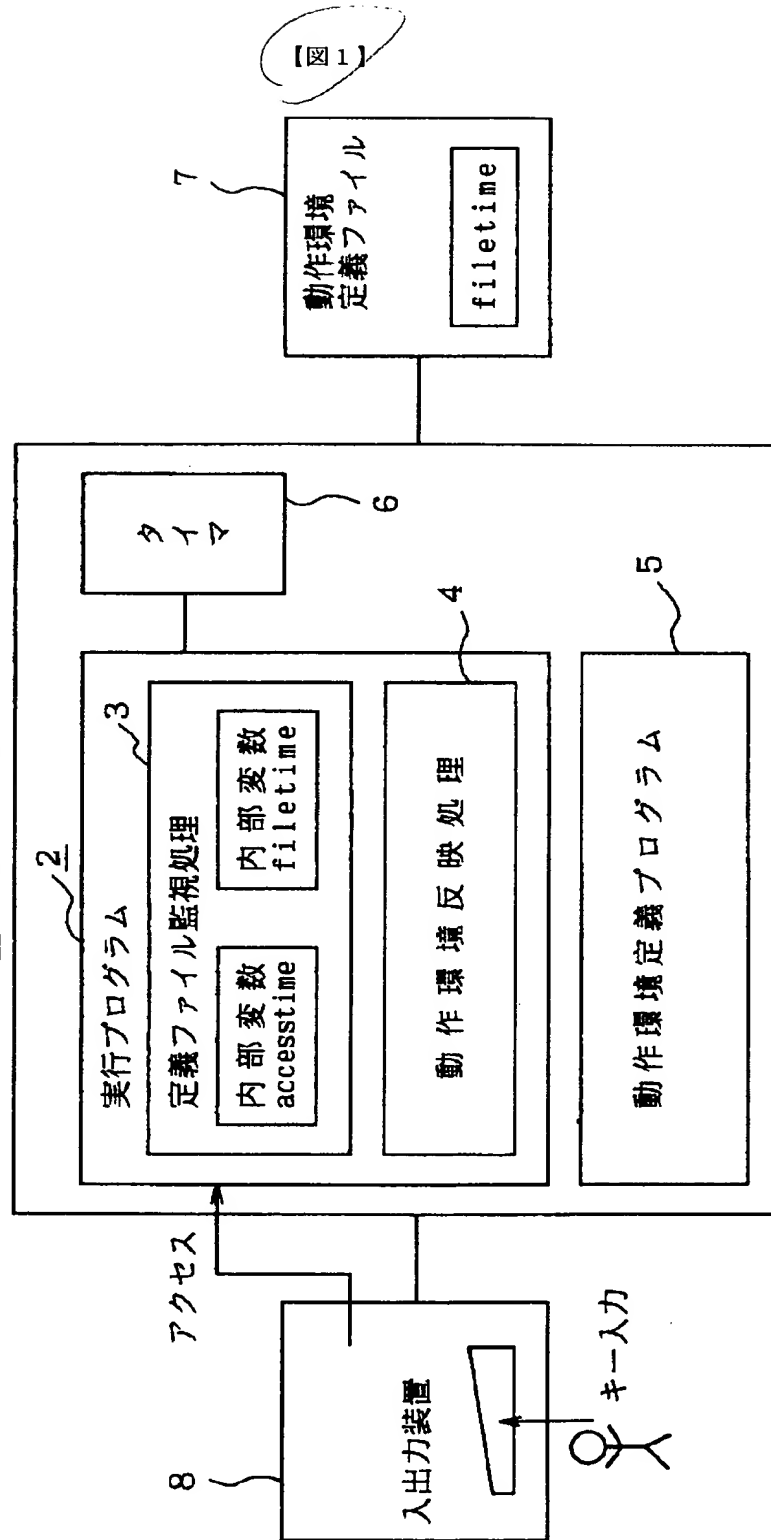
【図3】

本発明の動作環境定義ファイルの変更フローチャート



本発明の 1 実施例構成図

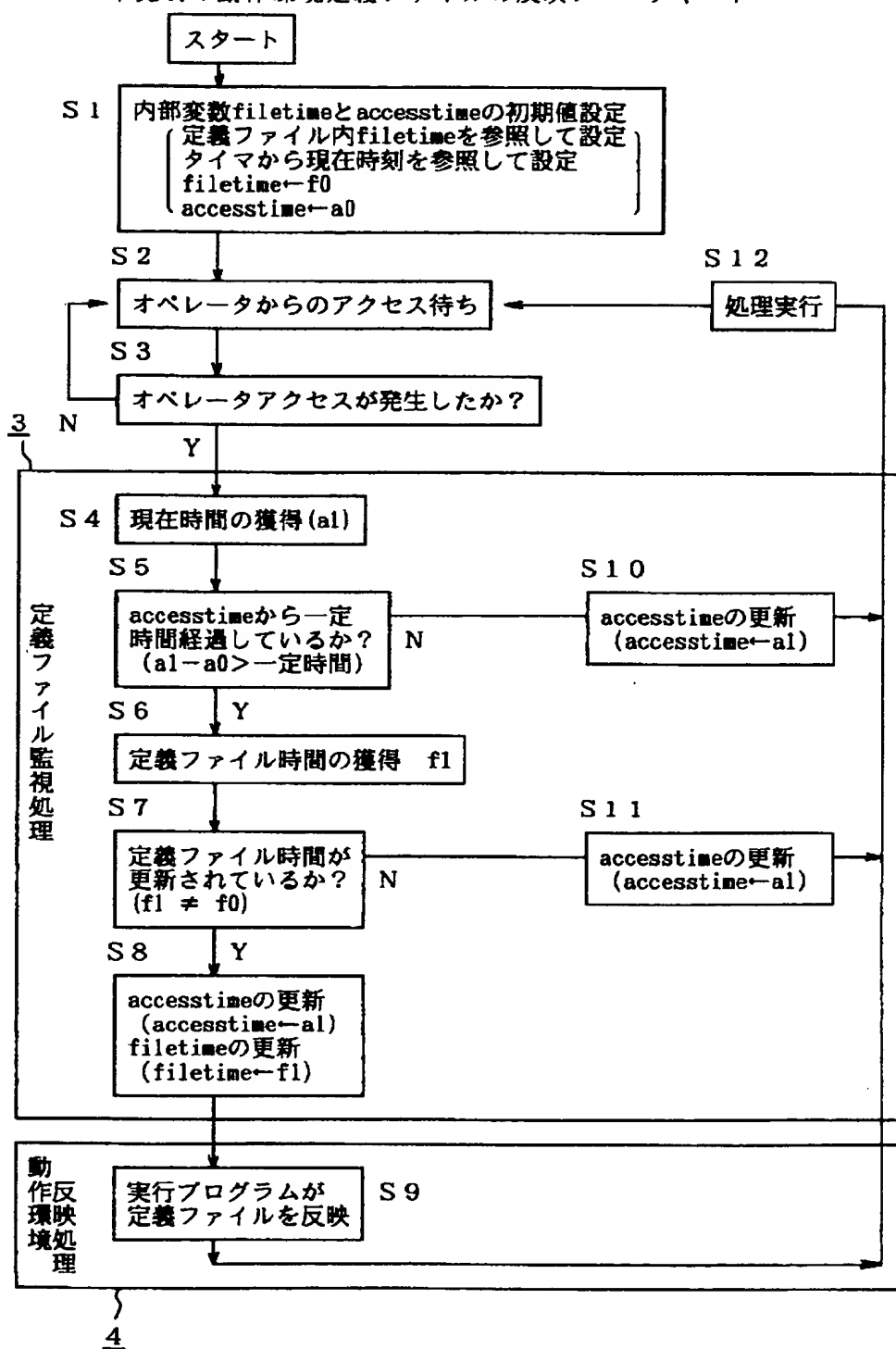
1: 演算処理装置



【図 1】

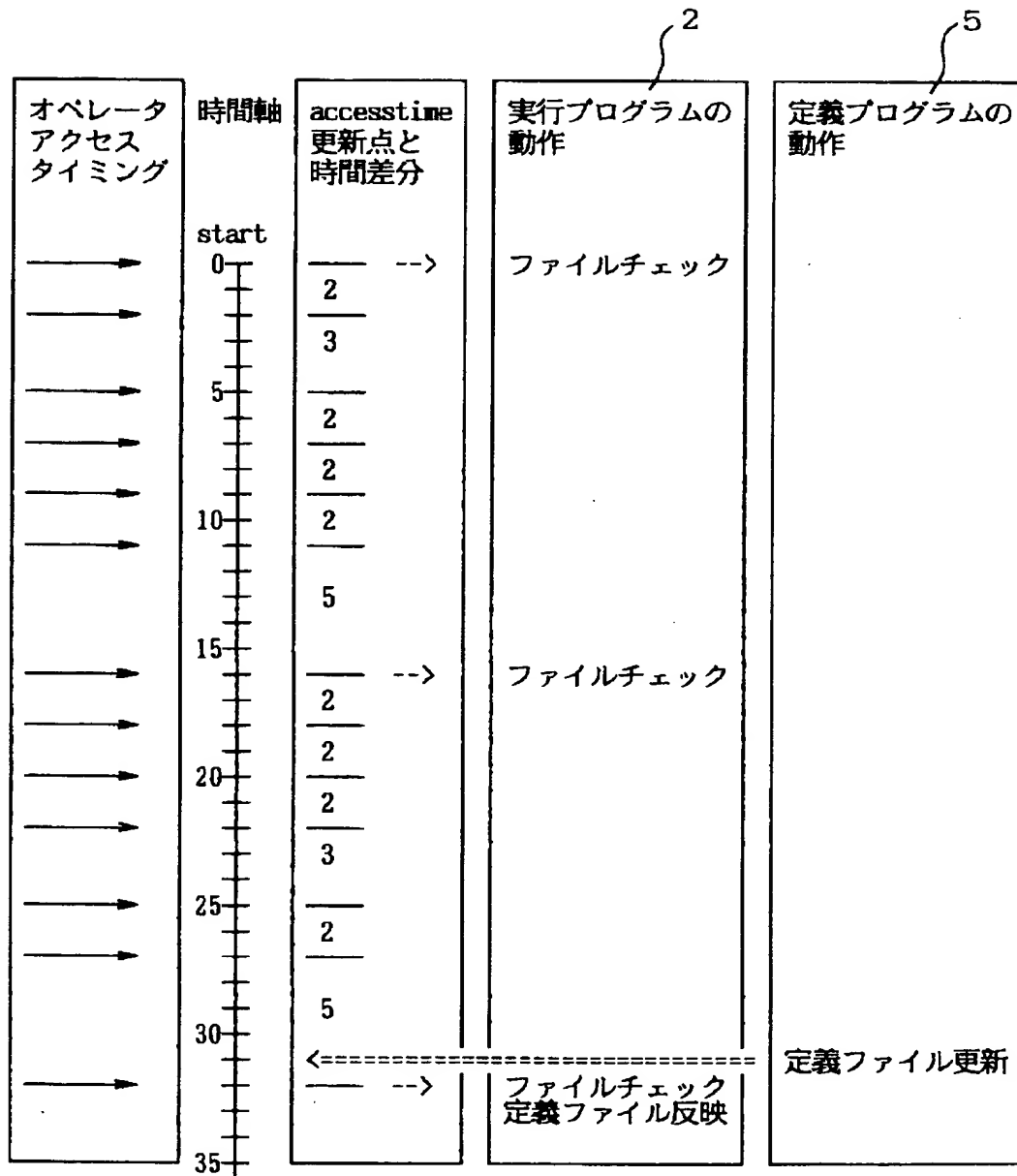
【図2】

本発明の動作環境定義ファイルの反映フローチャート



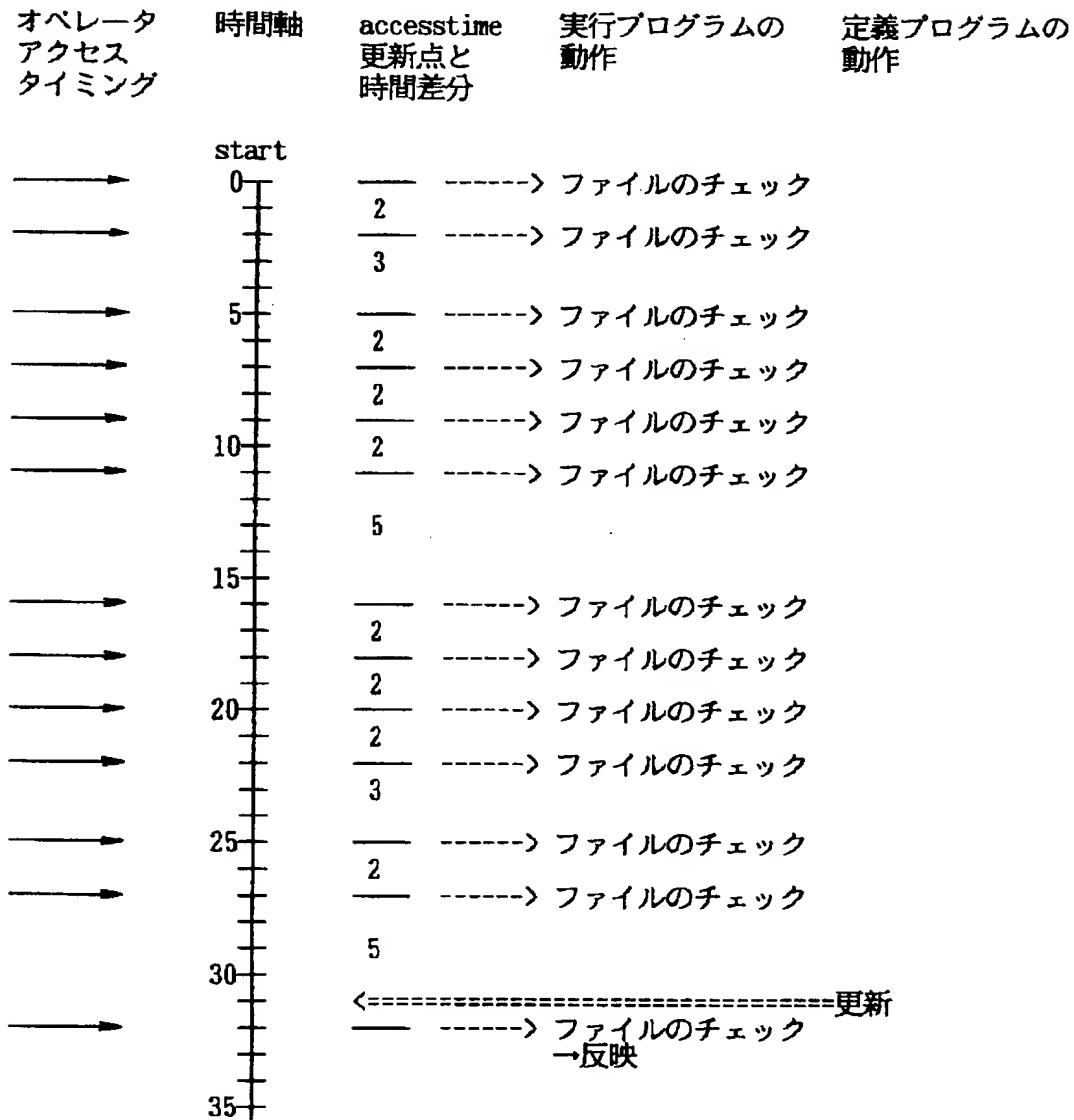
【図4】

本発明の具体例説明図



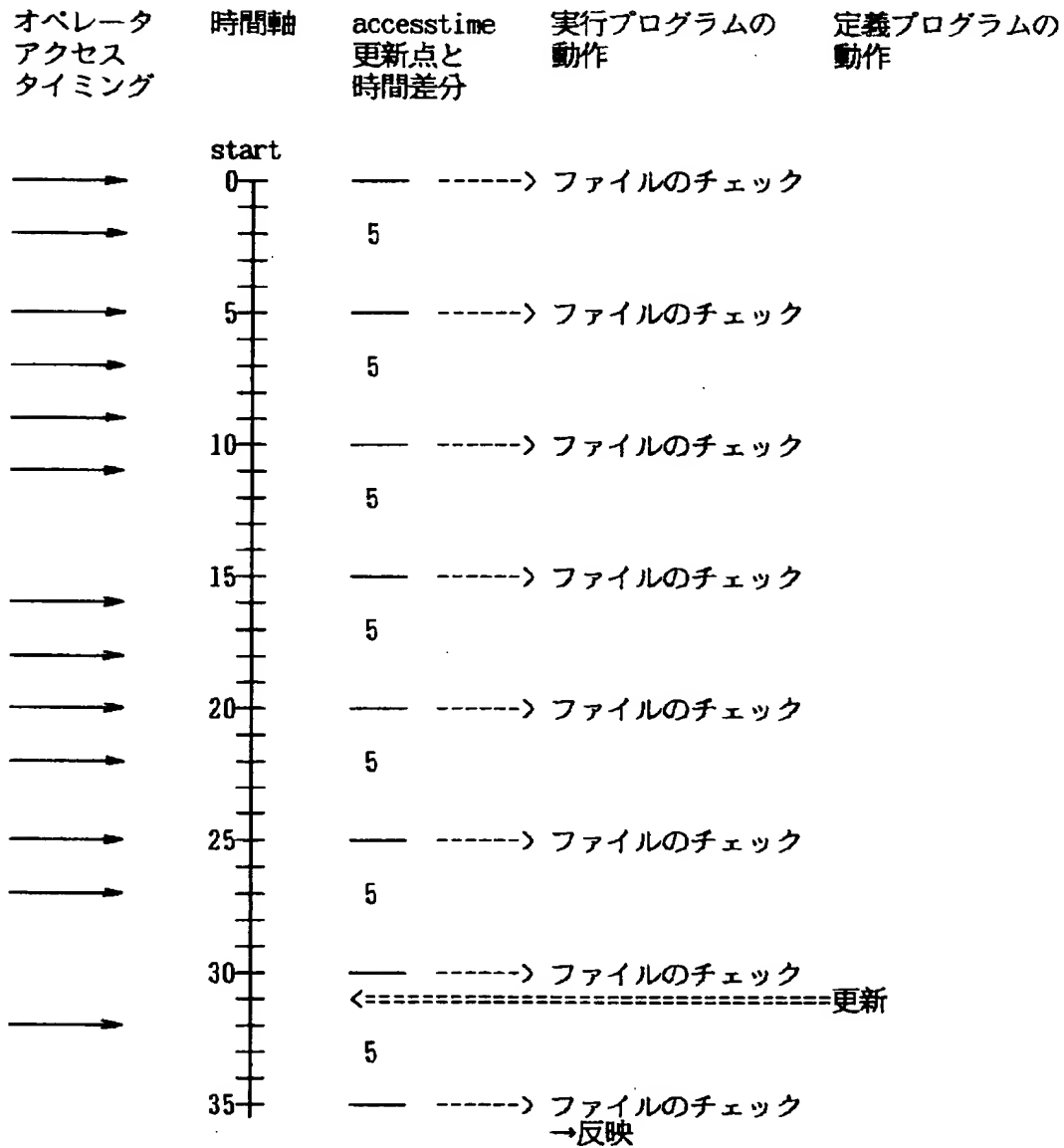
【図5】

従来技術の説明図（その1）



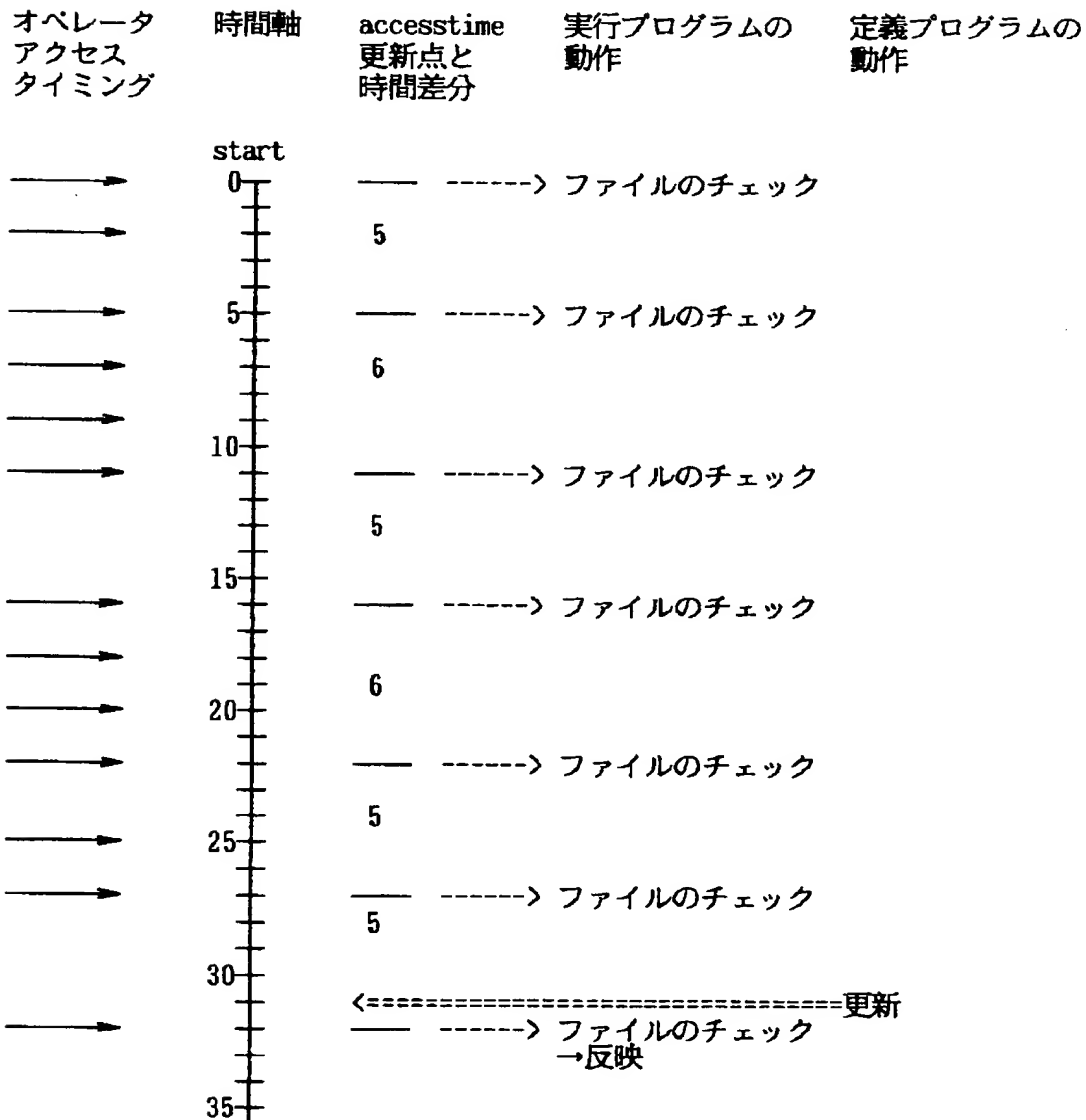
【図6】

従来技術の説明図（その2）



【図7】

従来技術の説明図（その3）



フロントページの続き

(72)発明者 渡利 亜紀子
 神奈川県大和市深見西四丁目2番49号 株
 式会社ピーエフユー大和工場内

(72)発明者 山田 利昭
 神奈川県大和市深見西四丁目2番49号 株
 式会社ピーエフユー大和工場内

(72)発明者 小池 匡
 神奈川県大和市深見西四丁目2番49号 株
 式会社ピーエフユー大和工場内

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.